Exercising a Native Intelligence Metric on an Autonomous On-Road Driving System

John A. Horst
National Institute of Standards and Technology (NIST)
100 Bureau Drive MS 8230
Gaithersburg, Maryland, 20899-8230 USA
Telephone: (301) 975-3430
Email: john.horst@nist.gov

Abstract— **The intelligence of artificial systems is well quantified by the amount of specified complexity inherent in the representation, provided we have tools to measure it. Some may generally agree with this claim, but argue that it is simply intractable to successfully and accurately measure the specified complexity of any system, no matter how it was represented. We respond to this important and substantive criticism by performing a computation required by the NIM on an example problem. We have chosen autonomous on-road driving, a problem that has already been solved by "systems" that are known to be both complex and specified, namely, humans. We will begin with a concise statement of the scope of the problem and a summary description of an appropriate system representation approach. We then apply a previously published Native Intelligence Metric (NIM) to measure the specification inherent in that representation and perform some preliminary intelligence measurements for a particular autonomous on-road driving subsystem. We claim that with an appropriate intelligence metric and an appropriate system representation, we can establish an equivalency between 1) the "state of the world" conditions, forming the input to the system, that the system can respond to successfully, 2) the system representation, and 3) the system performance. This equivalency is a potentially powerful result and is a key benefit and uniqueness of the theory proposed in this paper.**

I. Introduction

What constitutes a "good" system is a highly practical question. Answering it is at the heart of a productive economy. A substantial industry has arisen which exists solely to develop and apply metrics to common consumer products. In this light, measuring the "intelligence" of a system should largely be an attempt to formalize and extend this already common procedure. To extend system "goodness" metrics to system "intelligence" metrics will surely consist of metrics that also measure system adaptability, system learning, and system performance potential.

However, it is also well known that "simple" instinctual (*i.e.*, "hard-coded") behavior in living organisms possesses an elegance, efficiency, complexity, and adaptability that is far beyond the successes of today's artificial systems. Even instinctual system behavior is not well understood, since we cannot yet come near to reproducing it in artificial systems.

Furthermore, the formalization of almost anything in science starts with the simplest representations and moves to the more complex. This is our approach to intelligence metrics. Since instinctual systems can display remarkably complex and useful behavior, a formal intelligence metric should start with a representation that is one of the most analyzable. This is why we have chosen finite state machines as our representation method.

We also notice that current system intelligence metrics research is typically seeking to analyze system performance alone – regarding the system itself as a black box. The intelligence metric we propose is dependent upon, but not solely tied to system performance. Our goal is to develop a formal intelligence metric that is "native" or "innate" to the system. This means we can apply this metric to the system representation itself as well as to the system output.

Our approach seeks to be formal, seeks to measure native intelligence, and starts with an analysis of a representation particularly appropriate for instinctual system behavior. What might we gain from this approach? We claim that with an appropriate intelligence metric and an appropriate system representation, we can establish an equivalency between 1) the world conditions input to the system that the system can recognize, 2) the system representation, and 3) the system performance. This equivalency is a powerful result and is the key benefit and unique advantage of the theory proposed in this paper. This means that we can determine system performance if we possess either the recognized inputs or the system representation. Intelligence can potentially be measured without system realization! In the same way that system simulation allows engineers to efficiently and safely develop systems prior to costly physical realization, such an intelligence metric can offer similar advantages.

This intelligence metric that we have been alluding to has already been introduced in [1]. The actual content of the system code is measured by this metric. The metric is not independent of system performance, because the metric needs to know that the system contains the potential for demonstrably successful performance. However, the metric can be applied merely to a binary representation of the system design, as long as we can guarantee successful performance by the design

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Exercising a Native Intelligence Metric on an Autonomous On-Road Driving System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **National Institute of Standards and Technology (NIST),100 Bureau Drive,Gaithersburg,MD,20899**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
PerMIS?03, Performance Metrics for Intelligent Systems, 16-18 Sep 2003, Gaithersburg, MD

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**10**	

components.

In this paper, we present a system representation that is conducive to the application of the Native Intelligence Metric (NIM). Key to the success of such an application is that the system representation submit to formal analysis. Modularity and separability of behaviors are important as well.

II. The Native Intelligence Metric (NIM)

We define native intelligence as the *specified complexity inherent in the information content of a system*. This is reasonable since any artificial system can be expressed as a finite set of instructions encoded in some general purpose language together with a completely and independently known specification. By specification we mean some known connection to patterns, where patterns are things like specific instances of syntax, semantics, and useful functionality. For electromechanical systems, specification is, most importantly, useful functionality.

The NIM employs a chance elimination argument [2] to form a promising native intelligence metric for encoded systems. It consists of the following steps: 1) identify description of the intelligent system E of interest in some standard (analyzable) representation 2) identify the pattern D, which fully identifies the "specification" of the intelligent system; for an electromechanical system, "specification" is largely "useful functioning", 3) gather all the relevant "side information" I, which verifies that the pattern D is not "read off of" E, but is independent of E; I is also a complete description of the resources and information necessary to form the pattern D, 4) identify the system description D^* effecting D that delimits E, meaning that E "contains" D^* and that D^* is the portion of E that matches the pattern D, 5) define a chance hypothesis, H, that adequately describes the formation of D^* by random processes alone, 6) compute the probability, $P(D^*|H)$, that D^* might occur according to H 7) compute the complexity, $\phi(D|I)$, of generating the pattern, D, from I, 8) compute $-\log_2 P(D^*|H)$.

Both $-\log_2 P(D^*|H)$ and $\phi(D|I)$ form the NIM. The former indicates how much the system deviates from the chance hypothesis particularly, but from chance processes in general. This alone though is not sufficient to provide a measure of intelligence, for many highly unlikely events occur all the time, but D^* ensures that only the portion of E that effects useful functioning is conditioned by H. $\phi(D|I)$ is a measure of the complexity of of generating the useful functioning given I. We retain this portion of the metric in order to allow that an intelligent system may be extraordinarily difficult to construct, but perhaps showing a relatively smaller amount of useful functioning.

E is simply the representation for the particular system we wish to analyze. The system design needs to be in an analyzable format. A system expressed as a finite state machine (FSM) is such a format. How do we identify I and D? The relevant system specification will be captured in the pattern, D, and the side information, I. The gathering and identification of I and D is a challenging task for reasonably complex systems. The challenge lies mostly in quantifying useful functionality. We can easily underestimate the intelligence in the system if we miss or overlook key information. This constitutes a false negative, which will be very common. However, because complex specified information is typically highly differentiated from other complex specified information, false positives will be highly unlikely. System designers will want to ensure that the systems they design achieve the maximum (intelligence) value under the NIM. This is called "design for analysis" and is common now for system analysis via various performance metrics, particularly with consumer products.

III. RCS system representation approach

Our system representation consists of multiple, simultaneously executing FSMs, organized in the form of a tree. Each of these FSMs has modules relating to four functions: behavior generation (BG), value judgment (VJ), world modelling (WM), and sensory processing (SP). The approach is both an architecture and a design methodology (alternatively called RCS or 4D/RCS [3]). RCS offers a precise description of the structure and contents of the system (the architecture) as well as the sequential steps a system designer must perform to create a design (the methodology). Figure 1 is an example of an RCS architecture. RCS has already been tested successfully on a wide variety of large-scale control systems. The value of RCS lies in great measure in task and design complexity management and broad applicability.

The FSMs are simultaneously executing, which means the entire system has memory for multiple states. Furthermore, the FSMs form primarily a framework for computing, which allows for general purpose code to be used as state transition functions. Therefore, the system as a whole is a general-purpose computing machine.

In RCS, FSMs are the predominant vehicle for expressing system behavior. The RCS methodology is accompanied by design rules. The methodology is task-oriented in the sense that the natural characteristics of the task to be performed are what inform and dictate the structure and content of the system design. The tasks are expressed in the form of FSMs, arguing that any `if...then...else` structure in the behavior should be encoded as an FSM to gain overall system

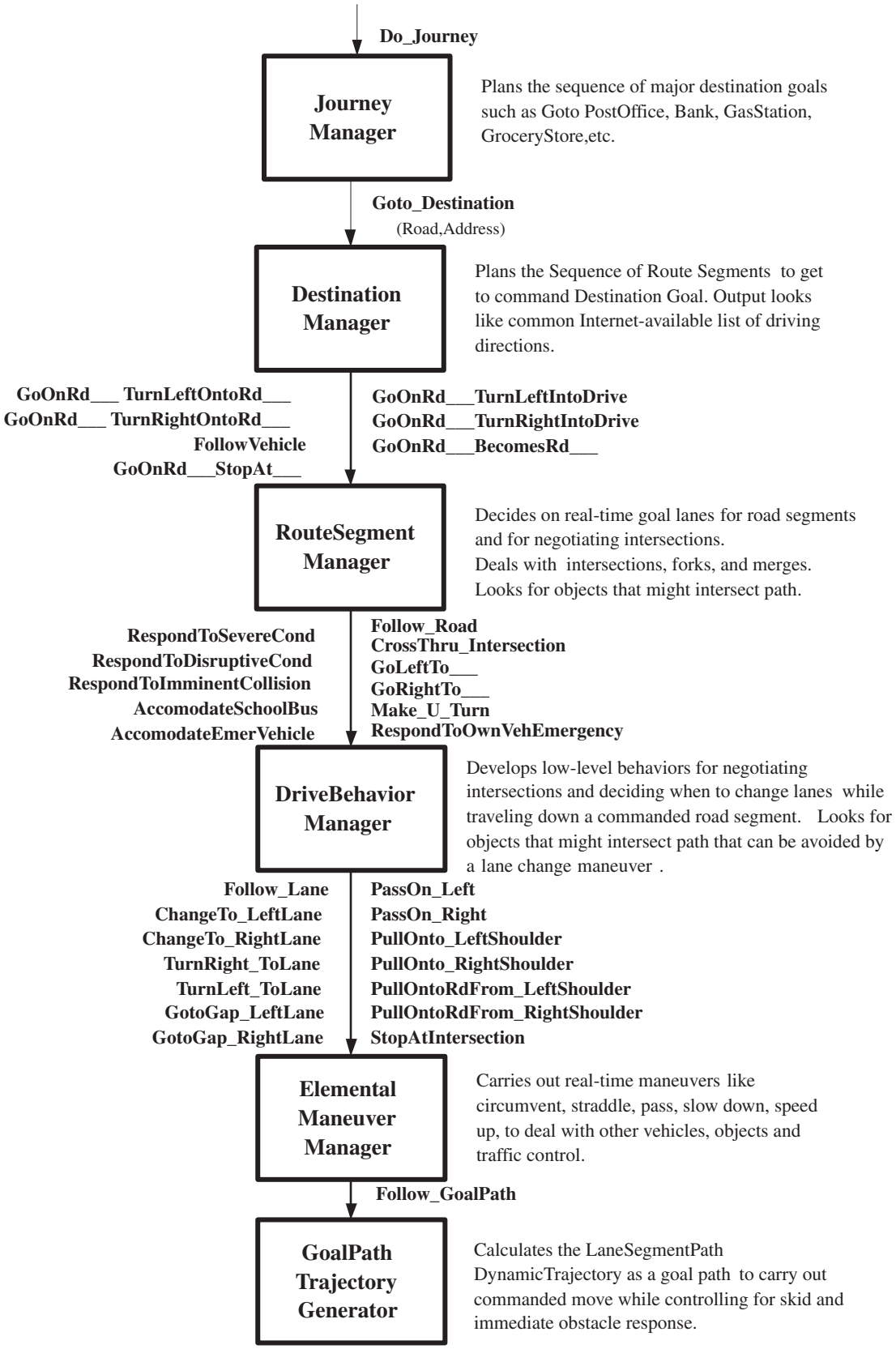

Fig. 1. RCS Control Nodes in the On-Road Hierarchy with Commands and Node "Job Descriptions"

clarity.

We now focus on the design method of RCS. It can be summarized as a design procedure and a set of design rules.

A. The FSM-based design procedure

1. Identify the highest level task associated with the application.
2. Precisely define the scope of the task including the identification of available resources, *e.g.* sensor, actuator, computing, and communication subsystems.
3. Break down higher level tasks into naturally constituent sub-tasks employing the design rules of Section III-B.
4. Perform this breakdown of tasks all the way down the hierarchy to the simplest and lowest level tasks.
5. For each task so generated, create an FSM, identifying all system and environment conditions (or decisions), functions, and data necessary for state transition. Each FSM formally expresses the synchronization and coordination of tasks for lower level FSMs.
6. Identify and select or create the necessary SP and WM primitive data and aggregate data needed to make the VJ decisions required for each node's state transitions.
7. Identify SP and WM data that needs to be shared inter-nodally.
8. Map nodes to processing habitats, *i.e.*, processes on specific processors, based on processing requirements.
9. Identify communication links between processors and set up communications.

B. Design rules

1. Separately defined tasks must be unique in the sense that they should have a substantial degree of behavioral dissimilarity in order to reduce redundancy and processing overhead.
2. The hierarchical decomposition of tasks should be designed based on the natural modular boundaries of each particular task and the unique nature of its environment.
3. Tasks and their FSMs should be defined in order to achieve an optimal number of states per task, since too few states increase system overhead and too many decrease system perspicuity, albeit the former is less important to adhere to than the latter, as long as good software design tools are available to handle overhead.
4. Tasks should be grouped into the same hierarchical level based on similar temporal and spatial scope of responsibility, with the caveat that what is actually performed by those tasks can be greatly dissimilar.
5. Nodes should be formed through the grouping of same-hierarchical-level tasks. Grouping of tasks into nodes is based on two rules.
   (a) The tasks should be behaviorally similar and support a definable "job description".
   (b) Tasks sharing the same sub-task must be grouped into the same node.
6. Tasks grouped into the same node must be mutually exclusive in time, i.e., two such tasks should not be able to be performed at the same instant of time; since tasks are realized as FSMs, this implies that each node is effectively a single FSM and therefore the entire system consists of multiple, simultaneously executing FSMs.
7. Employ non-mutually exclusive characteristics, such as global system conditions or certain environmental conditions, only to modify the FSM logic of mutually exclusive tasks and/or modify WM parameters within state transition functions in the FSMs of mutually exclusive tasks. This avoids combinatorial task explosion when non-mutually exclusive entities can easily be used to form a forbidding number of tasks.
8. Two (or more) parent controllers cannot command the same child controller at the same instant of time, implying that two parent tasks can be directly related to a single sub-task only if the two parent tasks are grouped into the same controller.
9. Tasks and sub-tasks should be defined to maximize reuse of sub-tasks.
10. Ensure that lower level tasks are grouped into a higher level tasks based on shared similarities of key task attributes.
11. Reveal or make explicit as FSMs all if..then..else activities in the system.

IV. The on-road driving application

The NIM can be applied to the on-road driving control application coupled with the RCS method of system representation. We will parameterize the scope of the problem and then conclude this section with an overview of how the NIM would be applied to a complex subsystem within this application.

A. Application scope

The vehicle is expected to be a common automobile, with minimal changes to allow for autonomous control, sufficient sensing, and on-board computing, *i.e.*, no radio link needed. No human interaction is expected at any point, i.e., complete autonomy is in view. Sensing will include auditory sensors, various types of cameras at various locations on the vehicle, including active (*e.g.*,

LADAR) and passive (*e.g.*, CCD) sensors, olfactory and touch sensors. All the computing required to achieve, goal-achieving control will be located on-board the vehicle. However, no adjustment of the external world will be assumed. Therefore, sign reading and comprehension, traffic light detection and comprehension, and two-way communication with humans, including the comprehension of signs and spoken natural language and some minimal natural speech output is required.

B. On-road driving and the NIM

We now proceed through all the steps of the NIM and describe roughly how subsystem intelligence can be measured by it. For simplicity, consider a single-obstacle maneuver subsystem, namely, a subsystem that has the ability to maneuver in-lane around a single slower-moving object. In Figure 1, the subsystem in view is the "elemental maneuver manager."

1) E is the description of the elemental maneuver manager subsystem in terms of BG FSMs, VJ decisions, and WM primitives and their aggregates. 2) Six possible vehicle in-lane maneuvers emerge, namely, circumvent object left, circumvent object right, run over object, straddle object, make contact with the object, and slow to follow object. D is the set of all these maneuver patterns for all possible types, sizes, dynamics of object. All patterns and object types fall into manageable finite sets. 3) Side information I is simply the set of driving rules that precisely describe the maneuver patterns D and the appropriate world states that trigger such patterns, and that such rules exist independently of D. If such a connection between I and D can be made, and D is of sufficient complexity, we can be assured that the maneuver patterns D are not due to a chance hypothesis. 4) D^* will be the portion of the subsystem design E that effects all the correct outcome maneuver patterns D, namely, those maneuvers that are appropriate for all possible world states. 5) The chance hypothesis H will simply be the proposal that all the appropriate maneuvers, D, generated by the subsystem D^* occurred by chance. 6) The probability $P(D^*|H)$ is the *probability* that the maneuver patterns D that are appropriate for each world state generated by the subsystem D^* occurred by chance. 7) Compute the actual complexity $\phi(D|I)$ of generating the maneuver patterns D given that designers know that rules of the road exist independently of the particular system. 8) Compute $-\log_2 P(D^*|H)$.

In summary, the more the subsystem D^* is able to generate maneuver patterns that exhibit efficient and safe behavior (such behavior as is detailed in drivers manuals and minds of good drivers) the lower will be the probability that such a maneuver subsystem D^* could have occurred unintelligently, or to put it positively, the higher the level of intelligence contained in the system. A key in this reasoning is that for E to be intelligent, I must be *detachable*. For maneuvering, it means that the concept of safe and efficient maneuvering existed before and, more importantly, independent of the design of system E. Negatively, this means that if safe and efficient maneuvering were *defined* based on the system we happened to build (i.e., D dependent on E alone), we would not be able to claim that E was designed based on intelligence and would be justified to claim that E was either incorrectly or haphazardly (or both) designed.

A key item then is to measure the amount of proper or appropriate maneuvering behavior for a wide variety of input conditions, namely, road conditions (hills, curves, surface), weather conditions (precipitation, visibility), obstacles and debris, and other traffic. This is difficult to quantify, but such quantifiable tests are constantly being performed on humans in driver's education courses. It is required that any system that is considered intelligent must have this quantifiable connection with a detachable pattern in the real world. For electro-mechanical systems this implies something like useful, purposeful, safe, and/or efficient functioning. Just how this quantity is measured is dealt with in the following section.

V. Computing $-\log_2 P(D^*|H)$ using the vehicle maneuvering example

The probability that the portion of the system responsible for the system E's useful functioning is due to a chance hypothesis is $P(D^*|H)$. If we were to compute this probability for a *purely random* hypothesis, we would easily obtain a vanishingly small number for any reasonably useful system; somewhat like the probability that a monkey with a computer and keyboard can type out Hamlet (Hamlet, in this example, is the system). We will still obtain very small numbers even for only partially random chance hypotheses, as we shall see.

Partially random chance hypotheses will not be useful for obtaining the *absolute* intelligence of the system, but since most measures of intelligence are eventually meant for comparison only, partially random chance hypotheses (relative chance hypotheses) will serve well. Besides, we suspect they will be substantially easier to compute.

Let's return to the problem of maneuvering in lane in the presence of an object[1] also in lane. We choose a relative chance hypothesis H with the following characteristics. When faced with an object in lane moving slower than the vehicle's desired speed, the vehicle will

[1]The object can be stationary or in motion; it can be a vehicle, pedestrian, or a rock.

uniformly randomly select one of the six possible maneuvers, namely, 1:circumvent around object left, 2:circumvent around object right, 3:run over object, 4:straddle object, 5:slow to follow object , and 6:stop on road shoulder. This is equivalent to a driver that, when faced with an object in his lane of a certain size, position, and velocity, rolls a single (fair) six-sided die and selects the behavior from the six possible behaviors based on the outcome of the roll of the die.

What is the probability, for a particular system, that the actual (or potential) behavior of the system occurred by this particular chance hypothesis? Assume for the sake of realism, the subsystem E_1 is currently unable to run over any object. Let's say E_1 cannot detect the *type* of object and therefore has no idea of the relative cost of that object, so E_1 will never run over an object, even if it is just a paper bag. Let's compare this with a subsystem E_2 that can perform all six maneuvers. Again to simplify matters, assume that, when there is a choice of several maneuvers, subsystem two just randomly chooses between the options. So does subsystem one, except that it never performs "run over object".

All possible $2^6 - 1 = 63$ combinations of allowable behavior for possible input conditions (input conditions are the various possible types, positions, and sizes of the object) need to be considered, where the number of one-, two-, three-, four-, five-, and six-maneuver conditions are 6, 15, 20, 15, 6, and 1, respectively ($6 + 15 + 20 + 15 + 6 + 1 = 63$). For example, three-maneuver condition sets are sets like {1,2,3}, {2,3,4}, {1,3,4}, etc. Since all possible conditions (events) are independent identically distributed (*iid*), individual probabilities multiply to compute the total probability for the j^{th} subsystem ($j = 1, 2$), $P(D_1^*|H) = \prod_{i=1}^{6} P(D_{j_i}^*|H)$ for the i^{th}-maneuver condition set.

The NIM requires that we compute the probability only for those portions of the subsystem that match with pattern, which for our subsystem means those portions that fail to match with pattern in some way are not part of the probability computation. Intuitively, this means that intelligence is both a function of what is both "correct" in the subsystem and what is not attributable to chance.

The world condition "able to run over object" will never cause subsystem one to actually run over the object, since it does not have that capability. So this world condition is not part of D_1^*. Therefore, the total probability for the 6 one-maneuver conditions for subsystem one will be $(1/6)^5$. On the other hand, subsystem two has the capability of detecting object type and so can run over the object if appropriate. This is pattern matching behavior and so is part of D_2^* and must be part of our probability calculation. So P("run over object"$|H) = 1/6$ for subsystem two. Therefore, the total probability for the 6 one-maneuver conditions for subsystem two will be $(1/6)^6$.

Continuing in this manner for all the other world conditions, for subsystem one, we exclude from the computation of total probability all the world conditions that do not have the match with pattern, namely, conditions that include "able to run over object". So when two maneuvers are possible (*i.e.* two-maneuver input conditions), 5 of the 15 sets of two-maneuver possibilities will contain the maneuver "run over object". Therefore, the total probability for the the two-maneuver conditions for subsystem one will be $(1/3)^{10}$ and for subsystem two will be $(1/3)^{15}$. Similarly, total probabilities for systems one and two for the 20 three-maneuver conditions are $(1/2)^{10}$ and $(1/2)^{20}$. Total probabilities for systems one and two for the 15 four-maneuver conditions are $(2/3)^5$ and $(2/3)^{15}$. Total probabilities for systems one and two for the 6 five-maneuver conditions are $5/6$ and $(5/6)^6$. The total probability for subsystem two for the 1 six-maneuver condition is 1.

This gives the total probability for subsystem one as $(5/6) \times (2/3)^5 \times (1/2)^{10} \times (1/3)^{10} \times (1/6)^5 \times 1 = 2.334 \times 10^{-13}$ and $(5/6)^6 \times (2/3)^{15} \times (1/2)^{20} \times (1/3)^{15} \times (1/6)^6 = 1.089 \times 10^{-21}$ for subsystem two. Finally, the NIM applied to subsystem one, $-\log_2 P(D_1^*|H)$, $= 41.962$ the NIM applied to subsystem two, $-\log_2 P(D_2^*|H)$, $= 69.637$. Our intuition about the additional intelligence of subsystem two is obviously satisfied. However, it may seem that the *amount* of the difference is unjustified. Perhaps we ought not exclude, from all probability calculations, all world conditions involving "able to run over object" for subsystem one.

VI. Autonomous on-road driving realization via RCS

We will now show how the system representation method (described in Section III) is being applied to the on-road driving control problem.

A. Hierarchy: nodes and job descriptions

In the design rules, summarized in Section III-B, we stated that the hierarchical decomposition of tasks should be performed based on the natural modular boundaries of each particular task and the unique nature of its environment. In this and other large-scale control problems, we find it remarkable how modular boundaries seem to arise so naturally as one ponders carefully the precise nature of the task and its environment. In several large-scale control tasks, namely, automated coal mining, dimensional inspection machine automation, and now autonomous on-road driving, this experience has been consistent [4].

For on-road driving, the task modularities have also been remarkably manifest. Referring to Figure 1, we see

that the highest level node, called Journey Manager (JM), handles trips from a home base to a home base. For example, our vehicle may need to stop at several destinations throughout our journey; we may go to work, stop at the store, visit a friend in the hospital, and return home. Decisions such as the ordering of the destinations are decided by JM. For example the store may close at 6PM, so if we are working late, we may need to visit the store before the hospital. The next level, called Destination Manager (DM), handles each destination. For example, going to the hospital may require us to follow route 97 north for 3 kilometers, go west on interstate 70 for 2 kilometers, go north on route 94 for 3.4 kilometers, and turn left onto Hospital Drive. DM is responsible for creating the kind of path plan which is currently available from certain internet sites, in which the plan may be adjusted based on criteria such as distance, time, or the route's scenic beauty. The Route Segment Manager (RSM) handles these route segments. Going north on route 97 might involve several intersections with traffic and obstacles in the road along the way. The RSM will process tasks that involve going from intersection to intersection. The next level, the Drive Behavior Manager (DBM), will handle driving behavior operations like passing and handling school busses and intersections. The next level, the Elemental Maneuver Manager (EMM), handles all in-lane maneuvers such as, circumvent object left, circumvent object right, run over object, straddle object, make contact with the object, and slow to follow object as described in Section IV-B. The subsequent levels handle trajectory generation and the parsing of global trajectories into the local trajectories and commands to a variety of servo level nodes including braking, steering, transmission, and throttle.

Each one of these nodes is an FSM. Actually it is an FSM framework, since general purpose computing (FSMs are not general purpose computing engines) can also occur within the node. However, the majority of behaviors occur in FSMs. The simplicity of FSMs gives potential gains in system and task perspicuity as well as the possibility of the formal analysis that FSMs allow. The currently executing task within each node forms a command to one of the subordinate nodes.

Hierarchical task decomposition modularizes task behavior, which greatly facilitates application of the NIM, as we will discuss in Section VII-A.

B. Behavior Generation FSMs (plans)

The heart of each node are the plans (FSMs) which realize the decision logic for each task within the node. In Figure 1 it can be seen that tasks to a subordinate node are sent as commands. A command received by a subordinate triggers the selection of a particular FSM from a set of preexisting FSMs. Such an FSM is shown at the top of the Behavior Generation column of Figure 2. In this example, a followLane command has been sent from the DBM node (the supervisory node) and received by the EMM node (subordinate). Based on conditions in the environment, different FSMs are spawned. For example, if the vehicle needs to manuever around two objects, both of which can be circumvented in lane but not straddled, an FSM called circANDnoStraddle1_circANDnoStraddle2, shown at the bottom of the Behavior Generation column of Figure 2, is realized.

One can see as well how these behavior generation FSMs fit into the context of the VJ decisions and the supporting WM primitives in Figure 2. We will discuss VJ and WM activity now.

C. Value judgment decision tree as transition conditions for behavior generation FSMs

The conditions of the environment that cause transitions in the behavior FSMs are generated in the Value Judgment (VJ) section of the node. These are the "conditions" on the left column in the state tables shown in the Behavior Generation column of Figure 2. The conditions that trigger specific behaviors depend in turn on lower level conditions that are also generated in VJ. The Value Judgment column of Figure 2 presents an example, for the followLane command in the EMM node, of the dependency of VJ conditions on lower level conditions. For in-lane maneuvering operations, vehicle dynamics, affecting weather conditions, road surface types, road characteristics, and in-lane obstacle conditions form some of the categories of conditions required for vehicle maneuvering behavior.

D. World model aggregation tree supporting value judgment decisions

The representation of the followLane command given in Figure 2 only reveals the contribution of primitive WM values. Actually aggregates of these world model values are used for (generally) higher level value judgment decisions. What is most important to point out is that this representation identifies exactly what world model primitives are needed to perform all the complex tasks described in Section VI-B. We will show later that with an FSM-based system, we can claim a formal equivalence between input streams of WM primitives, the particular FSMs employed, and the output system behavior. All this is connected to intelligence, because output behavior (performance) is connected to the input streams of WM primitives that the system "recognizes."

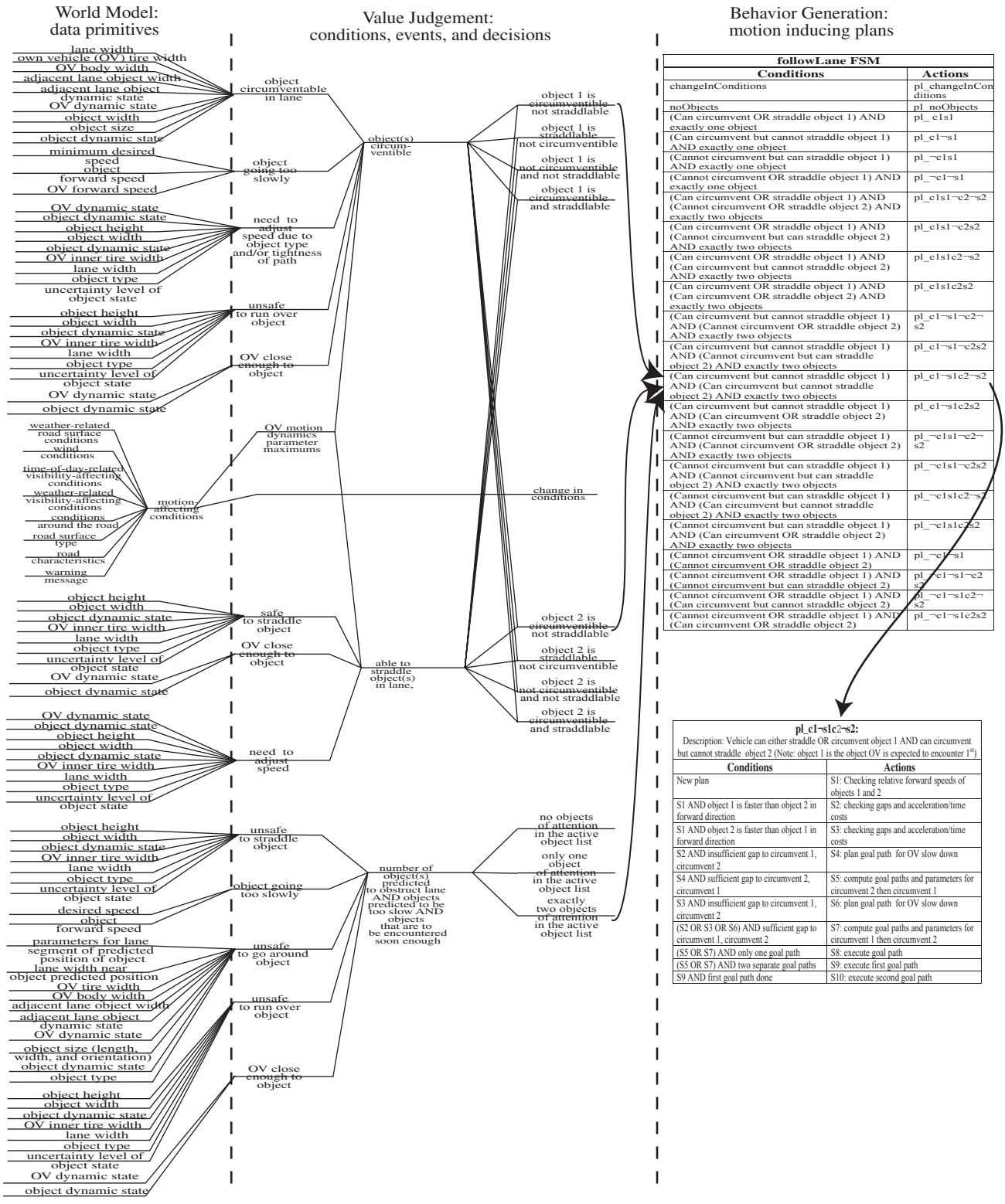

Fig. 2. Value Judgment decisions, World Model primitives, and behavior generation FSMs for followLane command. Arrows reveal the path from high level conditions to the followLane FSM to the pl_c1¬s1c2¬s2 FSM, where pl_c1¬s1c2¬s2 means "vehicle can circumvent both objects in lane, but cannot straddle either"

VII. Suitability of the RCS approach for NIM

A. Role of hierarchy

A hierarchy is helpful to the NIM for a couple of reasons. It enables the separability of behavior descriptions even though those behaviors operate in an integrated system. Separability is helpful also for reuse of components, but, more importantly for this work, allows independent measurement of component intelligence. Furthermore, a hierarchy is helpful to manage the combinatorics that naturally arise in state-based systems. This is important for the NIM, since it will allow one to simplify the computation of $-\log_2 P(D^*|H)$.

B. Role of behavior generation FSMs

Behavior as finite state machines exposes explicit and detailed states. The key is the separability where the addition of further behavior does not disturb other behaviors. The behaviors are largely uncorrelated, so that adding additional behaviors to the system does not substantively disturb the existing system. We suspect that this will have a similar benefit in the computation of the NIM, but such a question yet to be investigated.

C. Equivalence between input words and finite state automata

The "words" in our representation are the world model primitives that form the inputs to our system, and which come from the perception subsystem. The input words precisely match the system represented as an FSM. Finite state automata (FSA) theory reveals this to be so for the following reasons. FSMs in FSA theory are expressed as "decision problems" with two types of states, accept states and reject states [5]. An input string is either accepted or rejected if, at the end of processing the input string, the FSM is either in an accept state or a reject state, respectively. We say that A is the language of the machine, M, and write $L(M) = A$, where words, w, are in A. Furthermore, it can be shown that any FSM can be transformed into this simple accept/reject model, since decision problems can be transformed into computational problems in polynomial time [6]. Therefore, any FSM can be equivalently expressed as the set of input words it accepts.

This result implies that the set of WM state vectors of the intelligent system (the input words to the system) that the system recognizes is exactly equivalent to the behavior FSMs. But, in an *intelligent* system, the behavior FSMs produce *successful* behavior. If we define WM primitives over time as a string of words, $w_1, w_2, ..., w_n$, where the subscripts represent time and define successful behavior states as $S_1, S_2, ..., S_m$, then we have the following equivalency $w_1, w_2, ..., w_n \iff S_1, S_2, ..., S_m$.

How is the claim of equivalency helpful in applying the NIM to a system so represented? It places the system representation (summarized in Section III) in the domain of a formal theory. Any computational FSM can be reduced to a decision FSM, which opens up our system to formal analysis. For example, given any required WM input set, we can generate the equivalent decision FSM required to recognize it and *visa versa*. The WM primitive data needed for system behavior is precisely identified, providing guidance to the perception subsystem. Each accept state of the system must connect to demonstrably successful behavior. This is the side information and pattern, I and D, required by the NIM.

VIII. Summary and conclusions

The development and application of the NIM consists of the following items: the definition of and motivation for an appropriate NIM [1], the adoption of a system representation method that is amenable to measurement by the NIM, and a real-world application problem for experimental verification of the effectiveness of the NIM and the effectiveness of representation method to facilitate the application of the NIM. We have presented the second phase of this research, namely, the adoption of a system representation method and the start of the examination of the nature and effectiveness of the NIM for the on-road driving problem.

The key result of this phase of the research is that we have conceptually demonstrated an exact equivalency between the input the strings of input world model primitives that the system "recognizes," the FSM-based realization code for behavior generation, and demonstrably safe and efficient performance of the system. Though this may seem a trivial discovery, most intelligent systems metric research has been focussing on performance only, without exploiting this rich connection between WM data, BG code, and output behavior. Furthermore, with the modularity and uncorrelatedness of behaviors in the representation method (RCS), the formality, simplicity, and richness of FSM representation, and if we know the intelligence of subsystems, it then becomes possible to measure the intelligence of designs containing such subsystems, without directly measuring performance, though we must be assured of that the system can deliver successful performance.

FSMs, though providing opportunities for formal analysis, are quite simplistic and essentially less powerful than general purpose programming language, e.g., the Turing machine. However FSMs in this paradigm, though pervasive, merely are intended to form the framework for the intelligent system, and on-line planning, learning, and other general purpose operations can all be integrated into the framework.

What is the intelligence measured by the NIM of a sys-

tem designed according to this system representation? It is going to be proportional to the size of the accumulated set of either the input words "recognized" by the system, or, equivalently, the set of unique system "behavior states" that the system contains, where those states have been verified by external intelligent agents (humans) to produce demonstrably successful performance.

As we seek to build intelligent systems, a key to success is to achieve the concerted, integrated efforts of researchers and practitioners worldwide (lack of same has been a great hindrance). We cannot achieve such a concerted effort without an agreed-to "task taxonomy" for the tasks we wish to automate. For example, agreeing to a common organizational structure for military organizations greatly facilitated their efficiency and effectiveness historically. Once we have this in place for common tasks (like on-road driving)...and this is essential, though difficult...if we have a representation approach that is 1) transparent and 2) separable (i.e., can add intelligent behavior modules without disturbing existing behaviors), then we have the basis for a native intelligence metric, one that is not solely dependent upon system performance. We believe that the NIM will be critical for expediting system design, because it allows metrics to be fruitfully applied prior to connecting the system to simulation or actual realization with hardware in the real environment.

Because the NIM requires that the system have a measurable connection with useful, detachable, and complex functionality, we see that a NIM, rather than competing with performance metrics, will actually depend on them. We suspect that performance metrics will be applied to subsystem functionality and a modular, separable design method like RCS will allow us to build large scale autonomous systems that have globally measurable intelligence, based on the measurable performance of their subsystems.

Several research issues still need to be addressed. Real-time considerations are essential to the successful behavior of an electro-mechanical system, particularly an on-road autonomous vehicle. We need to consider how the FSM formality would handle temporal sensitivity.

It is also appropriate that the NIM is roughly like Shannon's information measure, in that it is the negative logarithm of a probability. We would like to investigate the richness of this similarity.

For the on-road autonomous driving application, as yet, we are still in the process of 1) generating the FSMs for the system and 2) verifying correct performance using simulation and animation.

There is some complication in the equivalency claim advanced in Section VII-C, given that our representation method is actually multiple, simultaneously executing FSMs, and each node is an FSM executing in parallel (effectively) with all other nodes. However, the principle is true for each individual node and the scaling up to multiple, simultaneously executing FSMs will have to wait for analysis in a later phase of this research.

References

[1] John A. Horst, "A native intelligence metric," Gaithersburg, Maryland, USA, 2002, perMIS 2002.

[2] William A. Dembski, *The Design Inference: Eliminating Chance Through Small Probabilities*, Cambridge University Press, 1998.

[3] J. S. Albus and A. M. Meystel, *Engineering of Mind: An Introduction to the Science of Intelligent Systems*, John Wiley and Sons, 2001.

[4] John Horst, "Architecture, Design Methodology, and Component-Based Tools for a Real-Time Inspection System," Newport Beach, CA, 2000, 3rd International Symposium on Object-Oriented, Real-Time, Distributed Computing (ISORC).

[5] Michael Sipser, *Introduction to the Theory of Computation*, PWS Publishing, 1997.

[6] *author correspondence with Dr. Meera Sitharam, University of Florida.*